# LearnJavaWeb

## Jsp

### JSP中的内置对象

在JSP中，Application= Servletcontext。

## MVC设计模式

浏览器将请求发送给servlet，servlet将请求中的数据封装在Javabean（request域）中，然后转发给jsp页面，最后返回jsp，显示给浏览器。

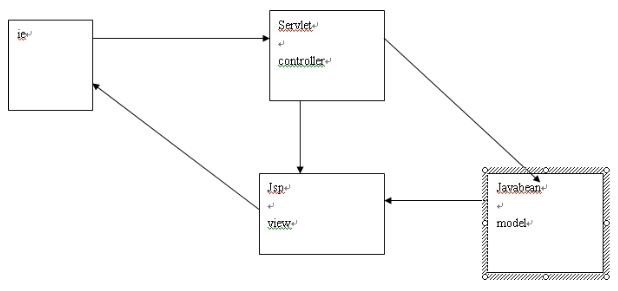


图 1 Jsp的MVC图示

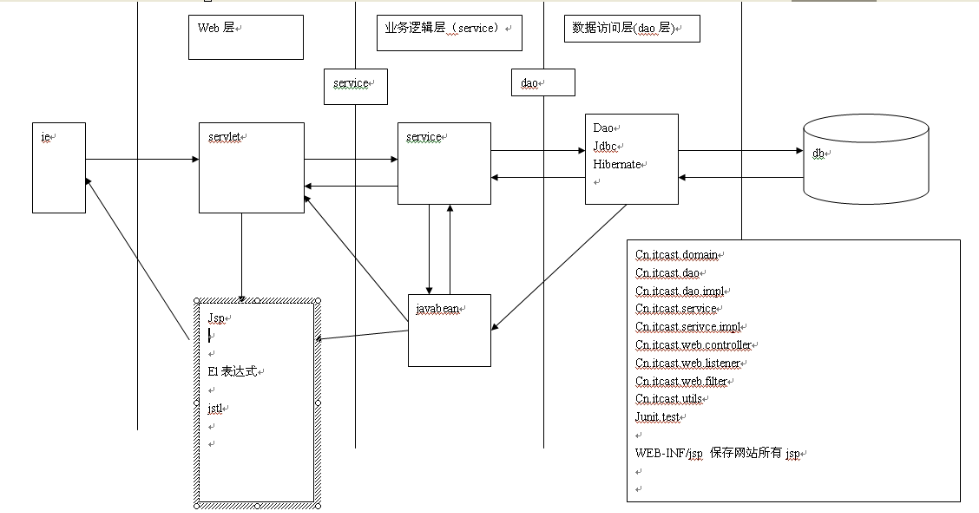


图 2软件开发层次示意图

## Strust

### Struct2项目配置步骤

### 自动注入

理解1：

先看一段代码  
假设你编写了两个类，一个是人(Person)，一个是手机(Mobile)。   
人有时候需要用手机打电话，需要用到手机的dialUp方法。   
传统的写法是这样:   
  
Java code   
  
public class Person{  
 public boolean makeCall(long number){  
 Mobile mobile=new Mobile();  
 return mobile.dialUp(number);  
 }  
}  
  
也就是说，类Person的makeCall方法对Mobile类具有依赖，必须手动生成一个新的实例new Mobile()才可以进行之后的工作。   
[依赖注入](https://www.baidu.com/s?wd=%E4%BE%9D%E8%B5%96%E6%B3%A8%E5%85%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWm1P9nhNBuynvnHRsPHF90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1b4rjTYn1mv)的思想是这样，当一个类(Person)对另一个类(Mobile)有依赖时，不再该类(Person)内部对依赖的类(Moblile)进行实例化，而是之前配置一个beans.xml,告诉容器所依赖的类(Mobile)，在实例化该类(Person)时，容器自动注入一个所依赖的类(Mobile)的实例。   
接口：   
  
Java code   
  
public Interface MobileInterface{  
 public boolean dialUp(long number);  
}  
  
Person类：   
  
Java code   
  
public class Person{  
 private MobileInterface mobileInterface;  
 public boolean makeCall(long number){  
 return this.mobileInterface.dialUp(number);  
 }  
 public void setMobileInterface(MobileInterface mobileInterface){  
 this.mobileInterface=mobileInterface;  
 }  
}  
在[xml文件](https://www.baidu.com/s?wd=xml%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWm1P9nhNBuynvnHRsPHF90ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1b4rjTYn1mv" \t "_blank)中配置依赖关系   
  
Java code   
<bean id="person" class="Person">  
 <property name="mobileInterface">  
 <ref local="mobileInterface"/>  
 </property>   
</bean>  
<bean id="mobileInterface" class="Mobile"/>  
这样，Person类在实现拨打电话的时候，并不知道Mobile类的存在，它只知道调用一个接口MobileInterface，而MobileInterface的具体实现是通过Mobile类完成，并在使用时由容器自动注入，这样大大降低了不同类间相互依赖的关系。

理解2：

IoC 或者 DI 或者 ...一大堆的缩写词  
不管是[面向对象](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWnyRLnvF9uh7BuWcLmyNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHnzrHD1nWR)，还是[面向过程](https://www.baidu.com/s?wd=%E9%9D%A2%E5%90%91%E8%BF%87%E7%A8%8B&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWnyRLnvF9uh7BuWcLmyNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHnzrHD1nWR)，都需要分成许多的块，然后由这些部件[协同工作](https://www.baidu.com/s?wd=%E5%8D%8F%E5%90%8C%E5%B7%A5%E4%BD%9C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWnyRLnvF9uh7BuWcLmyNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHnzrHD1nWR)完成任务   
要[协同工作](https://www.baidu.com/s?wd=%E5%8D%8F%E5%90%8C%E5%B7%A5%E4%BD%9C&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWnyRLnvF9uh7BuWcLmyNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHnzrHD1nWR" \t "_blank)就会产生依赖，一个方法调用另一个方法，一个对象包含另一个对象   
如果对象A包含对象B的话，就需要在A里new一个B   
[依赖注入](https://www.baidu.com/s?wd=%E4%BE%9D%E8%B5%96%E6%B3%A8%E5%85%A5&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dWnyRLnvF9uh7BuWcLmyNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHnzrHD1nWR)从具体类B里抽象出接口IB——IB的具体实现可能有很多B,B1,B2...很多种——这样A可以不用再new具体的B了，而是跟IoC容器说：我要一个IB（getBean("IB")）。然后，由容器根据配置文件来做具体的new的工作。具体new的是哪个，由配置文件从代码外部决定，要更换成B,B1,或是B2...修改配置文件就能做到，不用再改代码了

### 页面表单请求参数的注入到Action中的原理

在Struct2中，表单提交的数据（username）会自动注入到Action类中对应的属性（String username），前提是Action对象为这个属性提供了Setter（）方法进行注入。

### 注意事项

* 使用HTML的标签时，<form action=”userAction.action” />，需要.action。而使用struct2标签时，不需要<s:form action=”userAction” />

## Spring

### IoC

1、IoC(控制反转)

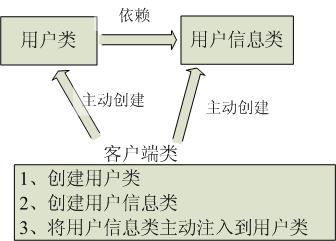


图 3 传统程序设计如图2-1，都是主动去创建相关对象然后再组合起来

首先想说说IoC（Inversion of Control，控制反转）。这是spring的核心，贯穿始终。所谓IoC，对于spring框架来说，就是由spring来负责控制对象的生命周期和对象间的关系。这是什么意思呢，举个简单的例子，我们是如何找女朋友的？常见的情况是，我们到处去看哪里有长得漂亮身材又好的mm，然后打听她们的兴趣爱好、qq号、电话号、ip号、iq号………，想办法认识她们，投其所好送其所要，然后嘿嘿……这个过程是复杂深奥的，我们必须自己设计和面对每个环节。传统的程序开发也是如此，在一个对象中，如果要使用另外的对象，就必须得到它（自己new一个，或者从JNDI中查询一个），使用完之后还要将对象销毁（比如Connection等），对象始终会和其他的接口或类藕合起来。

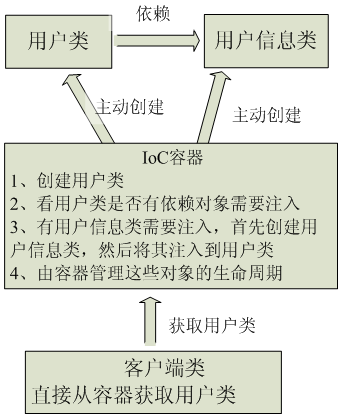


图 4 当有了IoC/DI的容器后，在客户端类中不再主动去创建这些对象了

　　那么IoC是如何做的呢？有点像通过婚介找女朋友，在我和女朋友之间引入了一个第三者：婚姻介绍所。婚介管理了很多男男女女的资料，我可以向婚介提出一个列表，告诉它我想找个什么样的女朋友，比如长得像李嘉欣，身材像林熙雷，唱歌像周杰伦，速度像卡洛斯，技术像齐达内之类的，然后婚介就会按照我们的要求，提供一个mm，我们只需要去和她谈恋爱、结婚就行了。简单明了，如果婚介给我们的人选不符合要求，我们就会抛出异常。整个过程不再由我自己控制，而是有婚介这样一个类似容器的机构来控制。Spring所倡导的开发方式就是如此，所有的类都会在spring容器中登记，告诉spring你是个什么东西，你需要什么东西，然后spring会在系统运行到适当的时候，把你要的东西主动给你，同时也把你交给其他需要你的东西。所有的类的创建、销毁都由 spring来控制，也就是说控制对象生存周期的不再是引用它的对象，而是spring。对于某个具体的对象而言，以前是它控制其他对象，现在是所有对象都被spring控制，所以这叫控制反转。

2、DI(依赖注入)

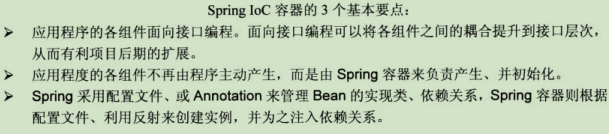
　　IoC的一个重点是在系统运行中，动态的向某个对象提供它所需要的其他对象。这一点是通过DI（Dependency Injection，依赖注入）来实现的。比如对象A需要操作数据库，以前我们总是要在A中自己编写代码来获得一个Connection对象，有了 spring我们就只需要告诉spring，A中需要一个Connection，至于这个Connection怎么构造，何时构造，A不需要知道。在系统运行时，spring会在适当的时候制造一个Connection，然后像打针一样，注射到A当中，这样就完成了对各个对象之间关系的控制。A需要依赖 Connection才能正常运行，而这个Connection是由spring注入到A中的，依赖注入的名字就这么来的。那么DI是如何实现的呢？ Java 1.3之后一个重要特征是反射（reflection），它允许程序在运行的时候动态的生成对象、执行对象的方法、改变对象的属性，spring就是通过反射来实现注入的。

理解了IoC和DI的概念后，一切都将变得简单明了，剩下的工作只是在spring的框架中堆积木而已。

在平时的java应用开发中，我们要实现某一个功能或者说是完成某个业务逻辑时至少需要两个或以上的对象来协作完成，在没有使用Spring的时候，每个对象在需要使用他的合作对象时，自己均要使用像new object() 这样的语法来将合作对象创建出来，这个合作对象是由自己主动创建出来的，创建合作对象的主动权在自己手上，自己需要哪个合作对象，就主动去创建，创建合作对象的主动权和创建时机是由自己把控的，而这样就会使得对象间的耦合度高了，A对象需要使用合作对象B来共同完成一件事，A要使用B，那么A就对B产生了依赖，也就是A和B之间存在一种耦合关系，并且是紧密耦合在一起，而使用了Spring之后就不一样了，创建合作对象B的工作是由Spring来做的，Spring创建好B对象，然后存储到一个容器里面，当A对象需要使用B对象时，Spring就从存放对象的那个容器里面取出A要使用的那个B对象，然后交给A对象使用，至于Spring是如何创建那个对象，以及什么时候创建好对象的，A对象不需要关心这些细节问题(你是什么时候生的，怎么生出来的我可不关心，能帮我干活就行)，A得到Spring给我们的对象之后，两个人一起协作完成要完成的工作即可。

　　所以控制反转IoC(Inversion of Control)是说创建对象的控制权进行转移，以前创建对象的主动权和创建时机是由自己把控的，而现在这种权力转移到第三方，比如转移交给了IoC容器，它就是一个专门用来创建对象的工厂，你要什么对象，它就给你什么对象，有了 IoC容器，依赖关系就变了，原先的依赖关系就没了，它们都依赖IoC容器了，通过IoC容器来建立它们之间的关系。

### IoC要点：



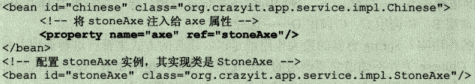
### 注入

设值注入

1. Spring通过xml来定义以来关系

<bean/>中的<property/>定义了类的属性，如果是基本数据类型，通过value即可对其赋值，而如果是其他类的话，通过ref=“bean\_id”类指定依赖的类，Spring会在调用无参数的构造器后（因此javabean类必需有无参构造器），创建默认的Bean实例后，调用对应的Setter方法为程序注入属性值。

程序通过id访问Bean，Bean与Bean之间通过id关联。



1. 在A类中依赖B类

只需在如下定义即可完成解耦，代码干净整洁：

|  |
| --- |
| Class A  {  //B为接口类，也是父类，利用多态，在XML中为其配有具体功能的子类，方便以后增加子类  private B b；  //为属性b设置setter()方法。  public void setB(B b)  {  This.b=b;  }  //无需自己new一个B类，Spring会在使用时通过xml找到A依赖的B，并通过反射创建B的实例，注入给A  public void useB()  {  b.fuction();  }  } |

构造注入

1. Spring通过xml来定义以来关系

<bean/>中的<constructor-arg/>指定了类在创建时需要的参数，如果是基本数据类型，通过value即可对其赋值，而如果是其他类的话，通过ref=“bean\_id”类指定依赖的类。

1. 在A类中依赖B类

只需在如下定义即可完成解耦，代码干净整洁：

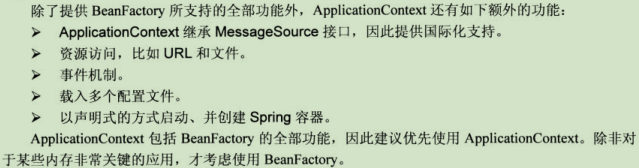
|  |
| --- |
| Class A  {  //B为接口类，也是父类，利用多态，在XML中为其配有具体功能的子类，方便以后增加子类  private B b；  //在构造器中依赖B类，spring为其注入  public A（B b）  {  this.b=b;  }  //无需自己new一个B类，Spring会在使用时通过xml找到A依赖的B，并通过反射创建B的实例，注入给A  public void useB()  {  b.fuction();  }  } |

1. 当A的实例创建完成后，B的实例创建完成，并注入到A中。

### Spring容器

BeabFactory与AppicationContext都可以表示spring容器，其中AppicationContext是BeabFactory的子接口。所有的Bean都由容器负责配置、创建、管理，bean可以是任何的java对象，java组件，包括数据源，hibernate的sessionFactory，事务管理器。

* BeabFactory实现类：XmlBeabFactory
* AppicationContext实现类：ClasspathXmlAppicationContext，FileSystemXmlAppicationContext

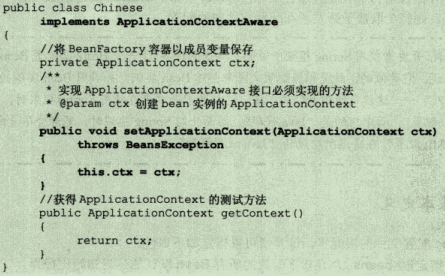




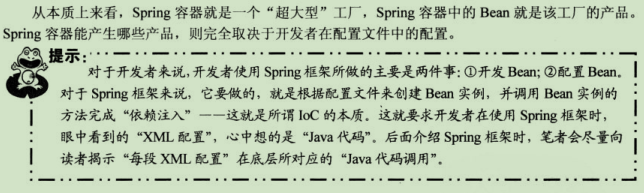
#### 获取spring容器

需要实现Beanfactoryaware接口，其中有一个方法：setBeanFactory（BeanFactory beanFactory），spring在创建该bean实例时，会自动调用这个方法，将引用指向自己。因此bean实例就可以获得创建自己的容器。

类似的还有ApplicationContextAware接口。



### Bean

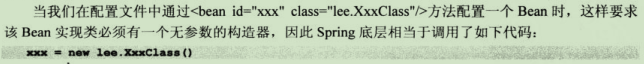


<beans>

<bean id=”” name=”” class=””> 有特殊情况时用name指定别名，代替id

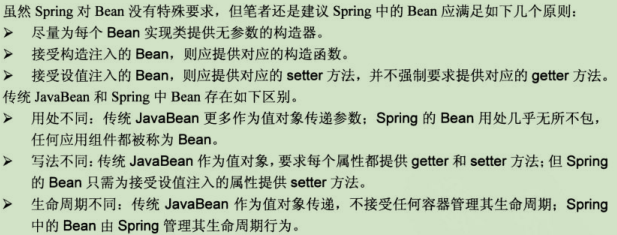
<bean/> 通过class指定编写好的bean类

<beans/>



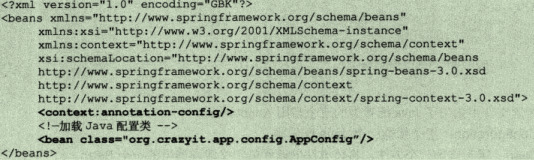
默认作用域为singleton，也就是单例模式，可以重复使用该实例。也可以通过scope指定，有如下5种：



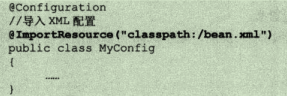


### 2种bean的配置文件

1. 以xml为主的，就要让xml配置能加载java类配置



1. 以java类配置为主的，要让java配置类能加载xml配置



### 推荐使用面向接口编程的原因：

Spring推荐面向接口编程，这样可以更好地让规范与实现分离，从而提供更好的解耦。对于一个java EE应用，不管是DAO组件还是业务逻辑组件，都应该先定义一个接口，该接口定义了该组件应该实现的功能，但功能的实现是由其实现类提供。

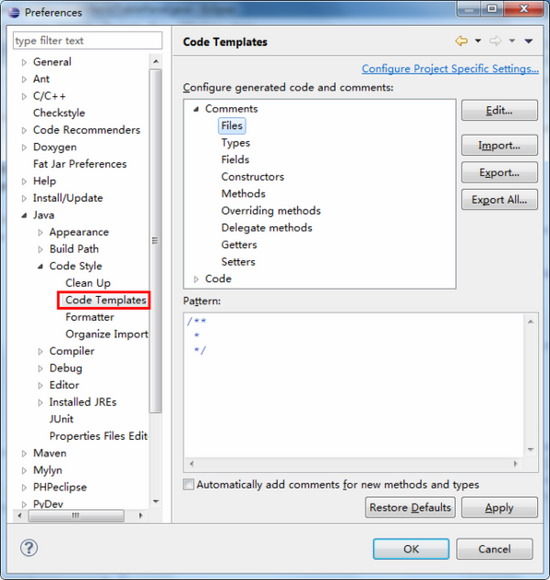
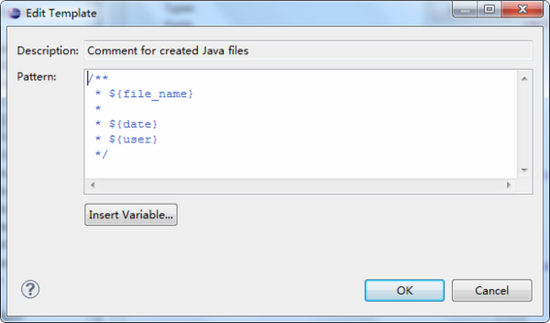
## Eclipse相关

### 函数添加注释说明

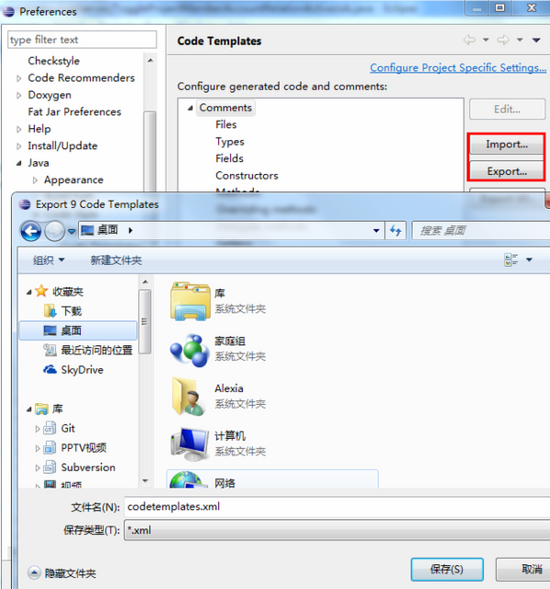
#### 快速注释说明

要把鼠标焦点放在方法前，然后ALT+SHIFT+J。

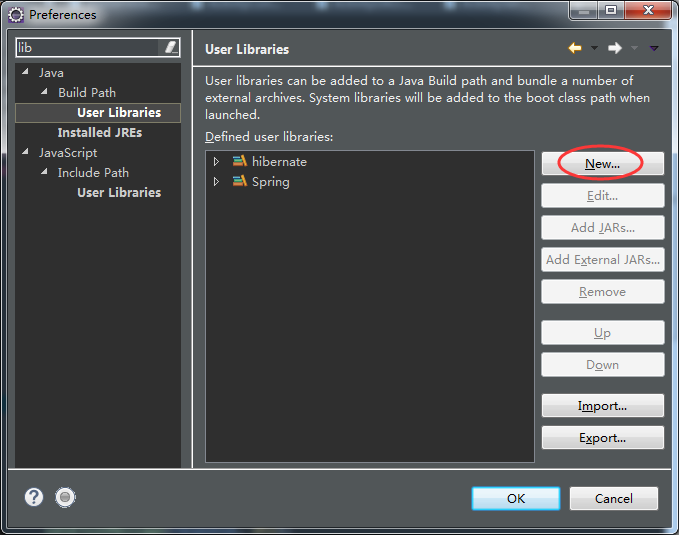
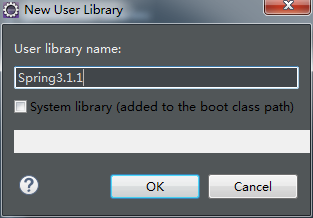
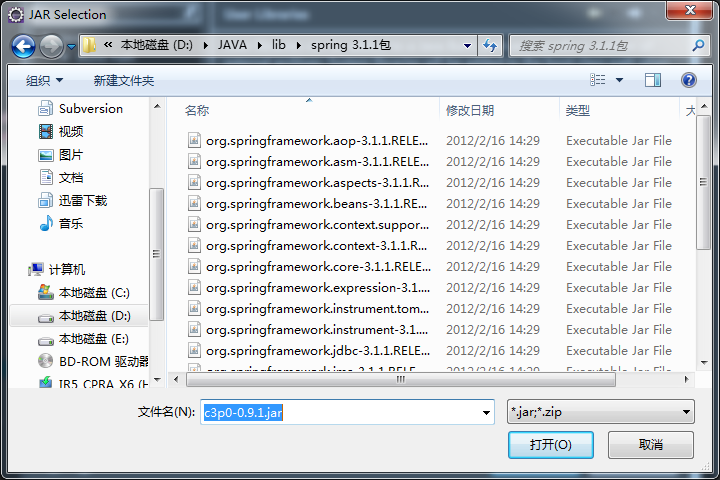
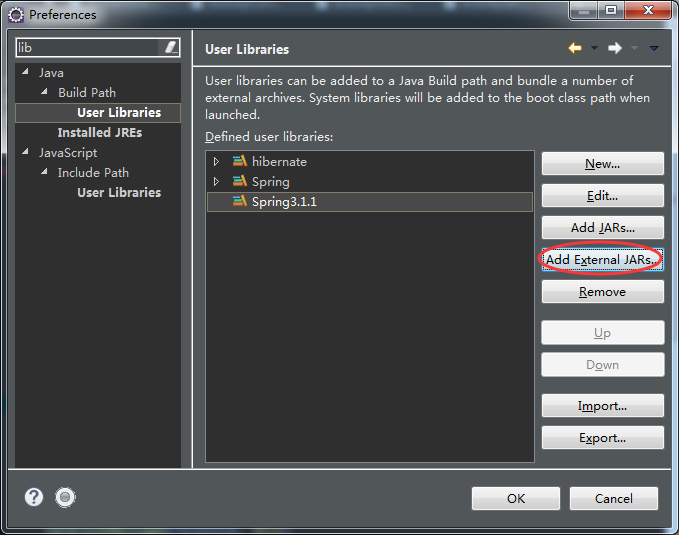
#### 自定义注释模板

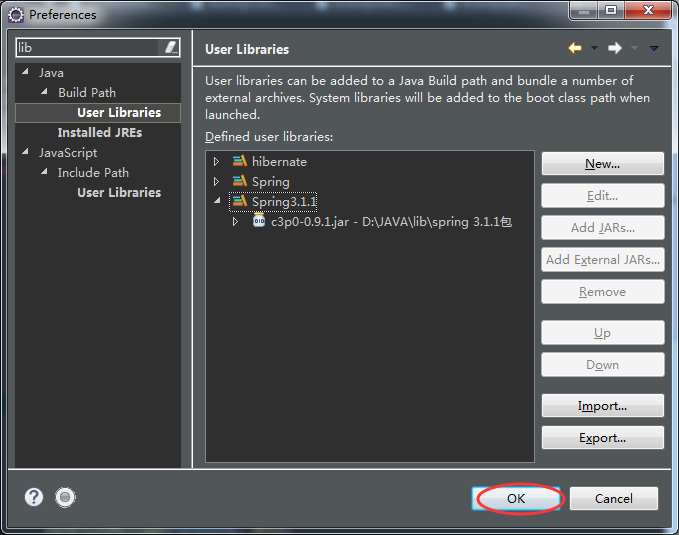
1. Window->Preference 打开参数设置面板，然后选择：Java -> Code Style -> CodeTemplates
2. 可以看到Comments下面一系列可以注释的Java代码组成部分（对文件、变量、方法、构造方法、重载函数等等），比如我想定制Java文件的注释风格，那么在右侧选择Comments中的File项，点击Edit，进入编辑模式，这样就可以自定义注释了。其中可以插入一些变量，如日期、文件名、作者等等。

#### 导入/导出代码格式模板

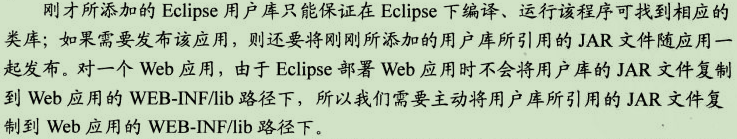
如果你辛辛苦苦定制好了自己的代码风格，然后换了台机器进行操作或重装了Eclipse，是不是要重新配置一遍呢？答案当然是No了，Eclipse提供了“导出”和“导入”功能，你可以把自己的模板导出来在其他机器上使用。

### 添加用户库（lib、JAR）

1. 点击window->preference，打开配置窗口，输入lib，即可快速找到，如下图：
2. 点击右侧的“new”，为第三方库命名，这只是命名还没加入jar文件  
   
3. 点击add external JARS，选择要添加的jar包  
   



注意事项：



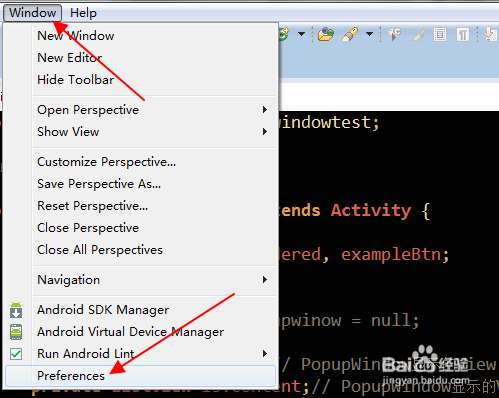
### 注释

|  |  |
| --- | --- |
| // 风格 | Ctrl+/ |
| /\*\*/ 风格 | Shift+Ctrl+/ |

### 乱码问题

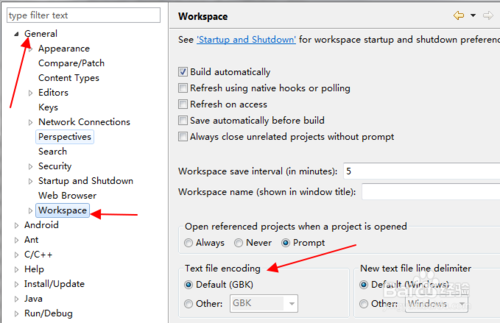
#### Eclipse默认编码

1. 编辑器的编码会影响到所有的项目中的字符的显示，可以说是作用最为广泛的设置，每一个项目都会受到这个设置的影响。点击菜单栏中的窗口（Window）— —选项（Preferences）。

[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=2)

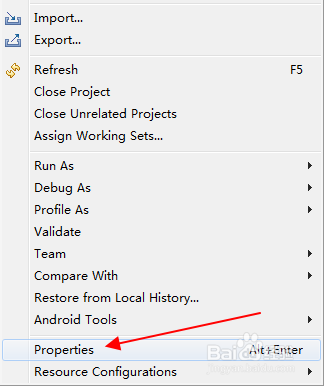
1. 2

点击常规（General）— —工作空间（Wordspace），然后在右侧的文本文件编码格式（Text file encoding）中选择你想要设置的编码格式，系统默认的格式为GBK，建议改为UTF-8

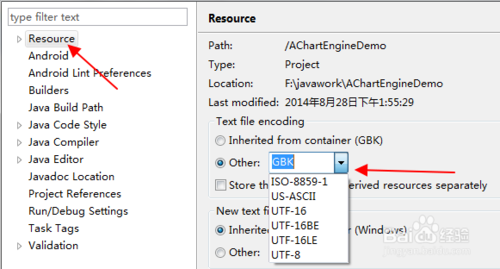
[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=3)

#### 整个项目编码

1. 如果其他的项目中不存在乱码问题，只是个别项目显示时出现乱码，那么我们不需要设置工作空间的编码，只需要修改项目的编码即可。
2. 选中项目后，右键点击，在弹出的菜单中点击属性（Properties）。

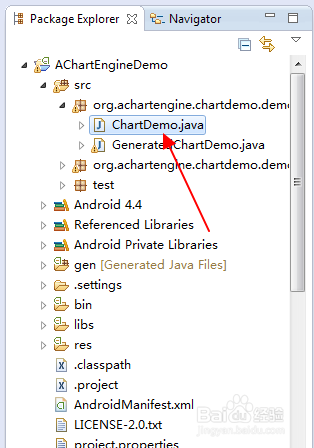
[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=4)

1. 3在弹出的属性窗口中，选择资源（Resource）— —点击选择Other（其他），然后选择你想要的编码格式。

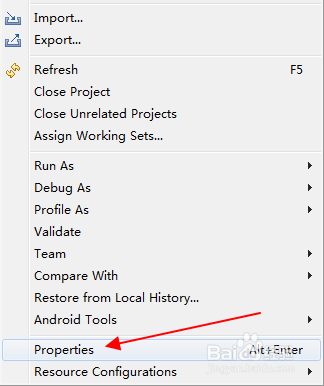
[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=5)

#### 单个文件编码

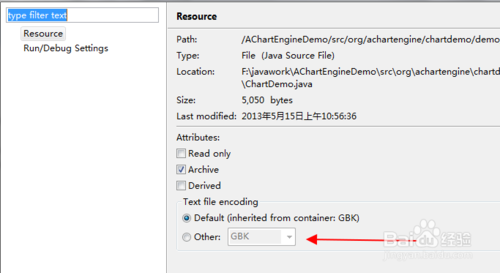
1. 只是个别文件出现了乱码，那么设置个别文件的编码格式就可以解决问题了。选中有乱码的文件，然后点击右键。

[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=6)

1. 在弹出的菜单中选择属性（Properties）。

[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=7)

3在弹出的属性窗口中选择资源，点击Other（其他）后设置个别文件的编码方式。

[](http://jingyan.baidu.com/album/870c6fc325a691b03fe4beac.html?picindex=8)